

Jacobs University Bremen

CO-522-B Communications Basics Lab

Fall 2021

Lab Experiment 1: Transmitter Simulink Study

Priontu Chowdhury

Introduction

The objective of this lab was to learn how a basic single-link communications system operates. The lab focused on high level simulation of the transmitter with Simulink, and gave us an understanding for what operations are required and what signals are involved. Important topics included the following:

- Bit to Symbol Mapping
- Constellation Diagrams
- Pulse Shaping Filters
- Up-conversion

The basic block diagram of a communications system is depicted below:

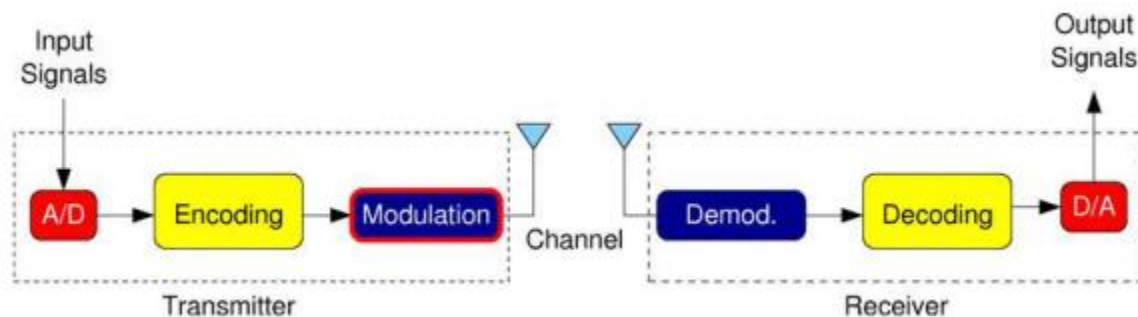


Figure 1: Basic Block Diagram of Communications System

In this lab, we focused mainly on the transmitter. The transmitter block diagram looks as follows:

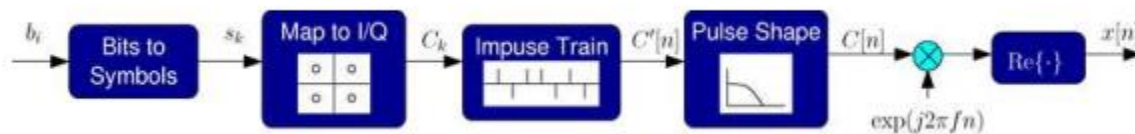


Figure 2: Basic Blocks for Developing a Transmitter

Symbol Mapping

We can typically transmit more than two different signals through a communications channel and detect which signal was sent on the other side. Consequently, groups of multiple bits are mapped to symbols to ease the process.

Constellation (I/Q) Mapping

I/Q mapping takes each symbol and maps it to a corresponding complex baseband in the complex plane. These values are chosen to be as different as possible to ensure that the receiver

can discern clearly what is transmitted. Constellation diagrams for different protocols are provided below:

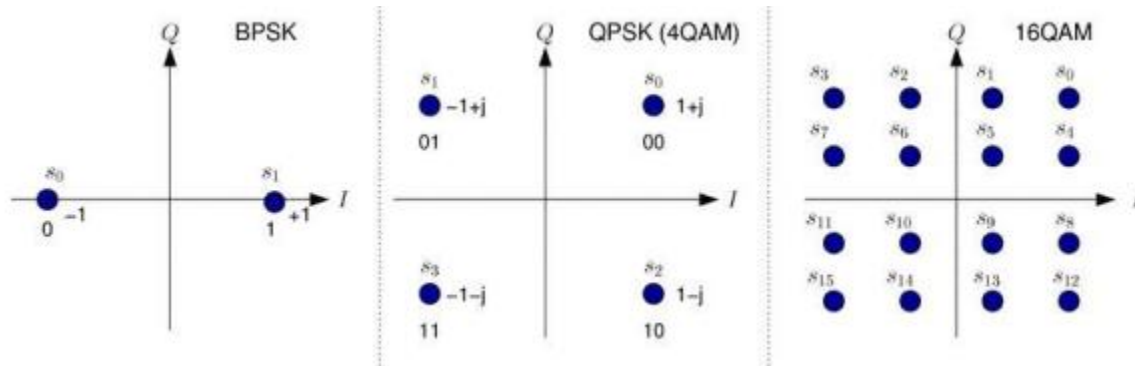


Figure 3: Constellation Diagrams for Different Transmission Systems

Pulse Shaping

Transmitting a rectangular pulse results in a sinc-like spectrum which is fairly wide, which is undesirable for applications like wireless transmission, where the bandwidth is limited. Instead, we can map each symbol to a pulse with a desired shape and transmit the superposition of these pulses in time.

We implement pulse shaping by converting the symbol stream to a train of pulses that are 1 sample wide and separated by the desired symbol period. By passing the pulse train through a filter that has the desired frequency response of our pulse, we obtain a set of shifted and superimposed pulses that are weighted by the desired I/Q values of the symbols.

Up-conversion

Since we do not transmit signals in baseband often, we need to shift the resulting complex baseband waveform to a higher frequency, which is achieved by the final multiplication block.

Execution

We used DSP Blocks to set up Simulink first. Then, we carried out the series of tasks outlined in the lab manual.

1. Starting Simple:

For this part, we connected the input stream to the scope and ran the simulation. We obtained the following setup:

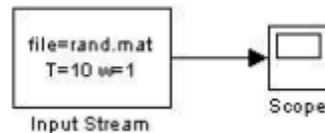


Figure 4: Simple Set-up

The scope results look as follows:

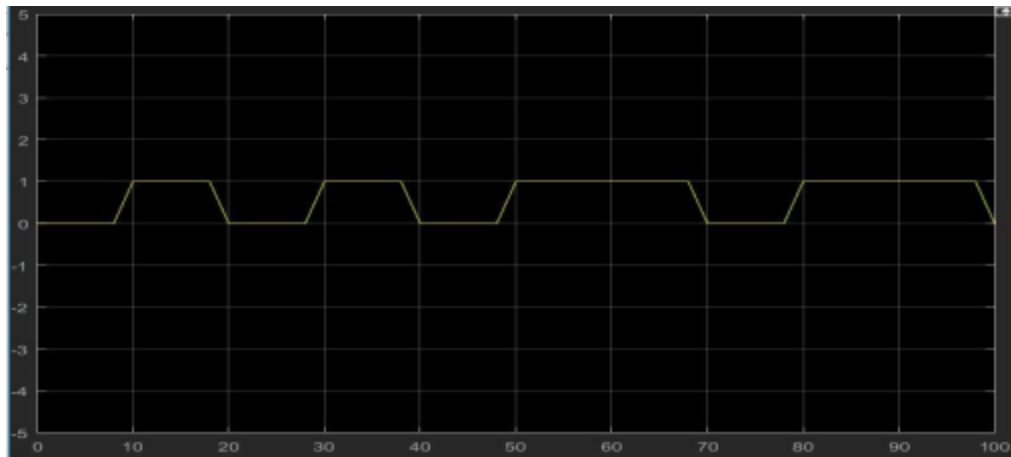


Figure 5: Input Stream

2. BPSK Symbols:

We started with a simple module, created with the input stream block from the DSP Blocks and a Scope. The set-up is as follows:



Figure 6: BPSK Symbol Setup

The data is received from an input stream, from which it is passed through a “Bits to Symbols” converter, which is then passed through a “Real Lookup 1D” block, and the result is presented on the scope. The symbol mapper has the parameter $[-1, 1]$.

On running the system, we obtain the following results on the scope:

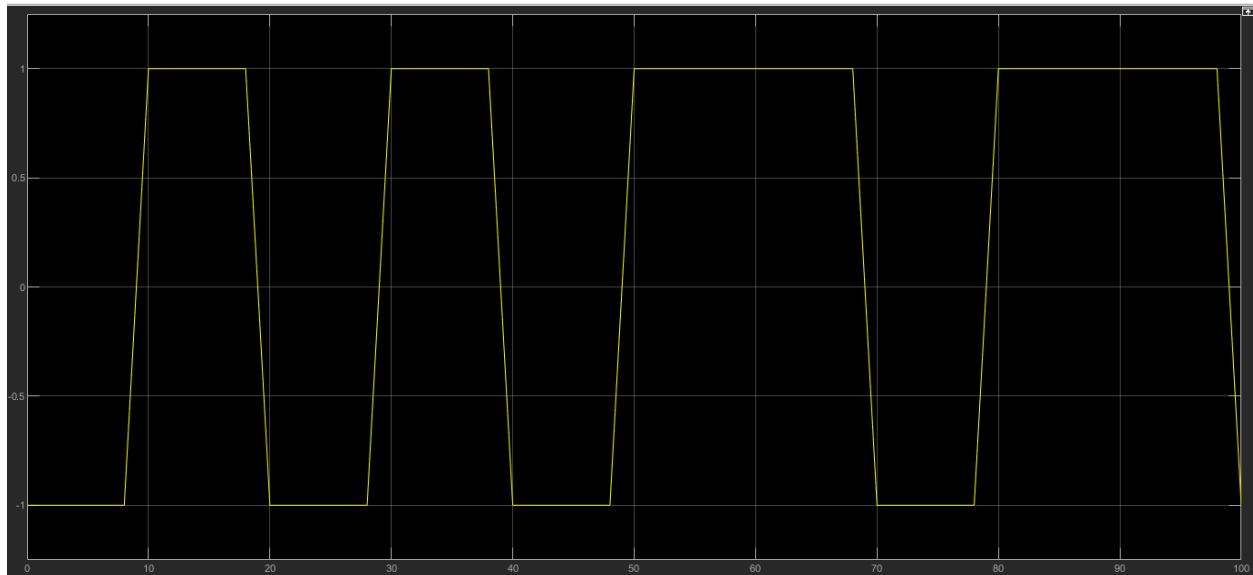


Figure 7: Output on scope on running a simple system

It can be seen that the output is either 1 or 0, which means we are getting random bits, with one new bit per 10 sample period.

3. Rectangular Pulses and PSD:

On following the instructions, we have the following system:

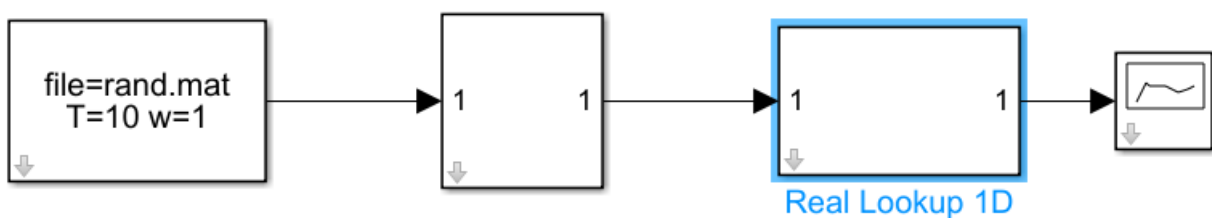


Figure 8: BPSK Symbol Setup

For this part, we replaced the scope with the PSD, provided PSD parameters as shown in the manual, and changed the simulation time to 10,000. On simulating the system, we obtain the following results:

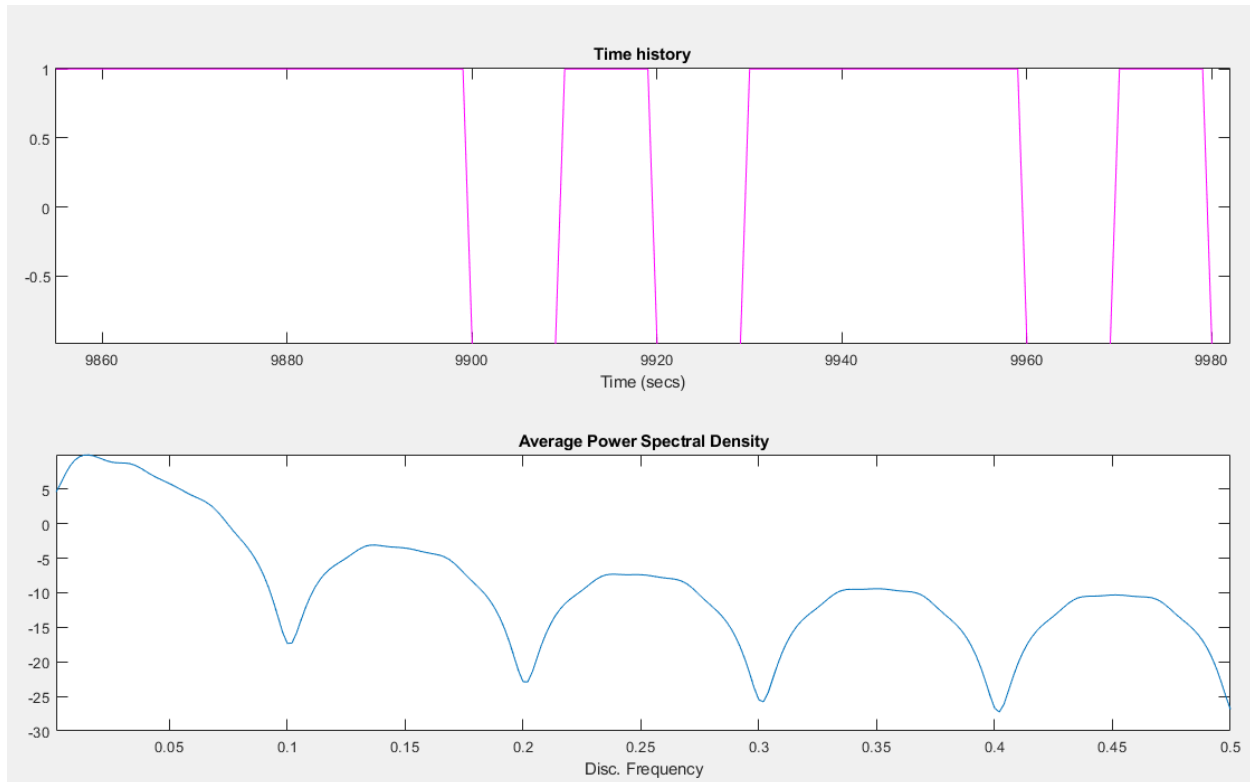


Figure 9: Simulation results on PSD

The PSD operates by taking the FFT of the slices of the input stream and averaging the results.

4. Spectrally-Efficient Pulse Shaping

We introduce pulse shaping in this part in an attempt to improve the pulse shape. The setup looks as follows:

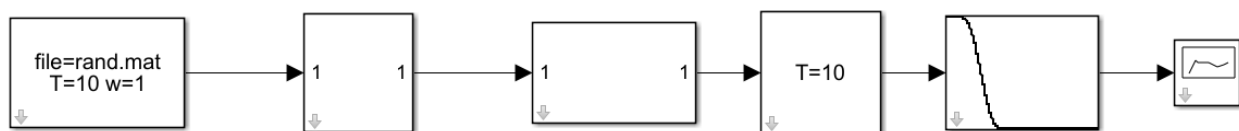
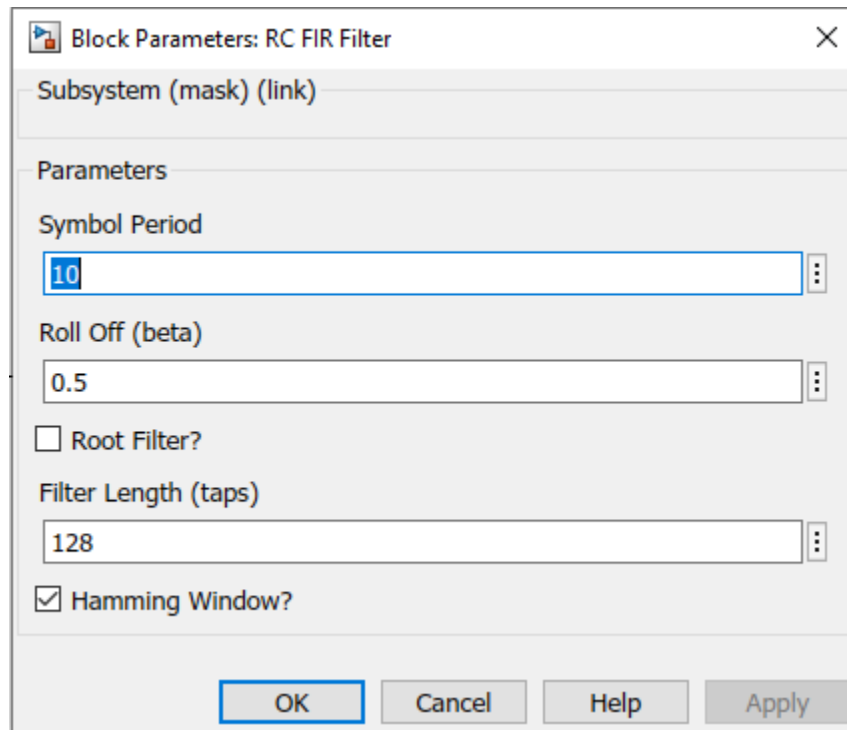


Figure 10: Setup for Spectrally-Efficient Pulse Shaping

The pulse train samples its input once per period and generates just a single unit impulse. Feeding this into the pulse shaping filter then generates a series of pulses. We used the Raised Cosine (RC) filter in this lab.

Settings of RC filter are the following:



The image shows a MATLAB/Simulink dialog box titled "Block Parameters: RC FIR Filter". It contains the following settings:

- Subsystem (mask) (link): (empty)
- Parameters:
 - Symbol Period: 10
 - Roll Off (beta): 0.5
 - ☐ Root Filter?
 - Filter Length (taps): 128
 - ☒ Hamming Window?

Buttons at the bottom: OK, Cancel, Help, Apply.

Figure 11: RC Filter settings

The results of simulation are as follows:

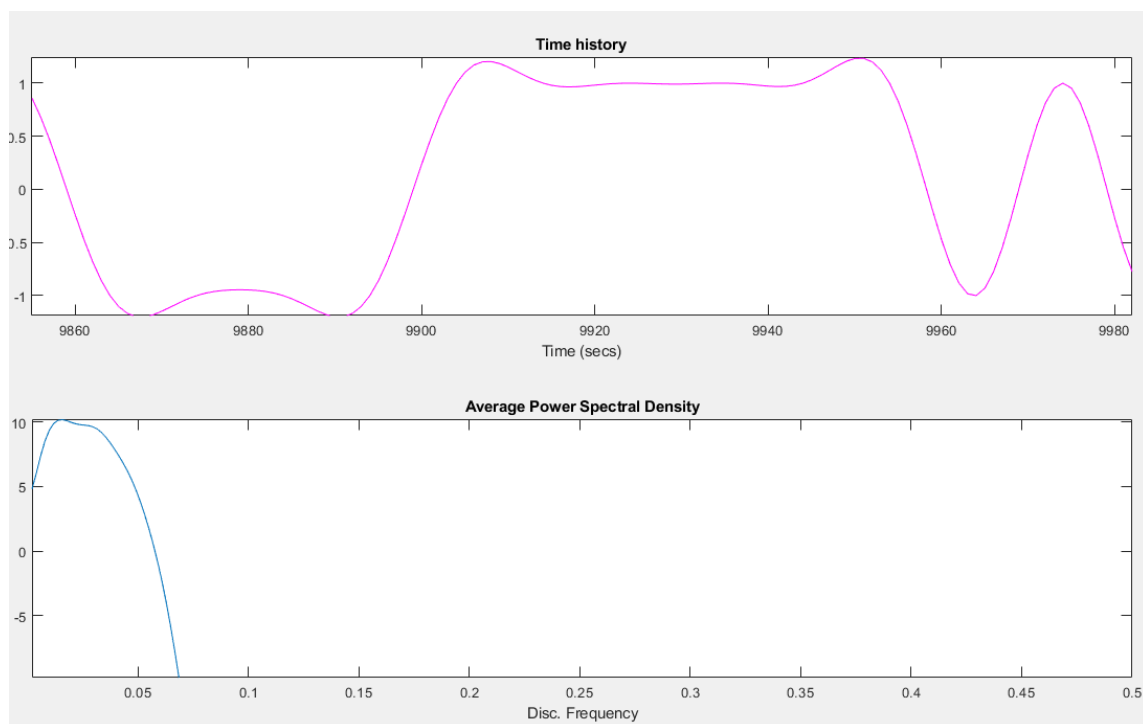


Figure 12: Simulation Results for Spectrally-Efficient Pulse Shaping Task

5. Decoding Pulses

On this task, we focus on comparing the pulses of the rectangular and the pulse-shaped waveforms. To compensate for the extra delay added by the filter, a delay block was used. The setup looks as follows:

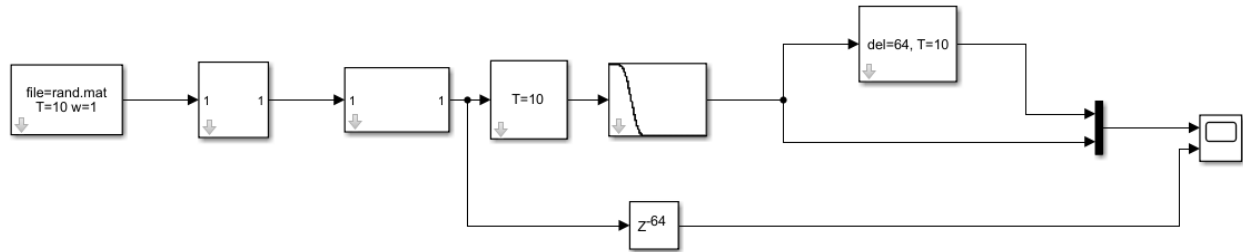


Figure 13: Setup for Decoding Pulses

After simulating, we obtain the following results on the scope:

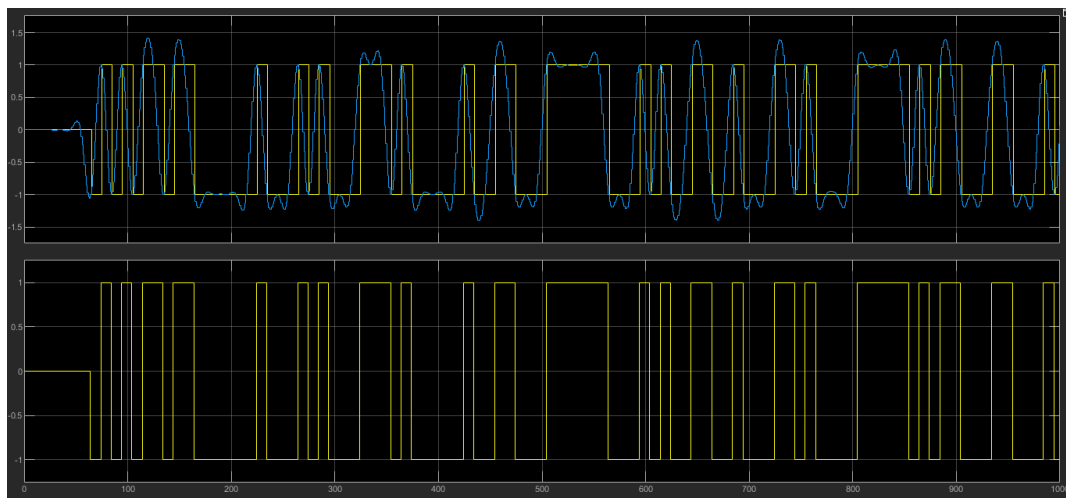


Figure 14: Simulation results for the above setup

The first plot shows the combination of incoming waveform and sampled waveform, and the bottom plot shows the original digital stream.

6. Up-Conversion

It is important that the signal is centered about some carrier frequency. To accomplish this, the pulse shaped waveform is multiplied with the sine carrier wave ($f = 2 \cdot \pi \cdot 0.5 = \pi$ Hz). The setup looks as follows:

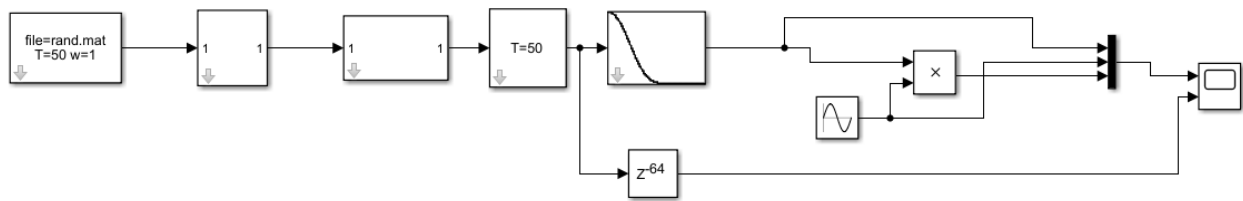


Figure 15: Setup for Up-conversion

The simulation results are provided below:

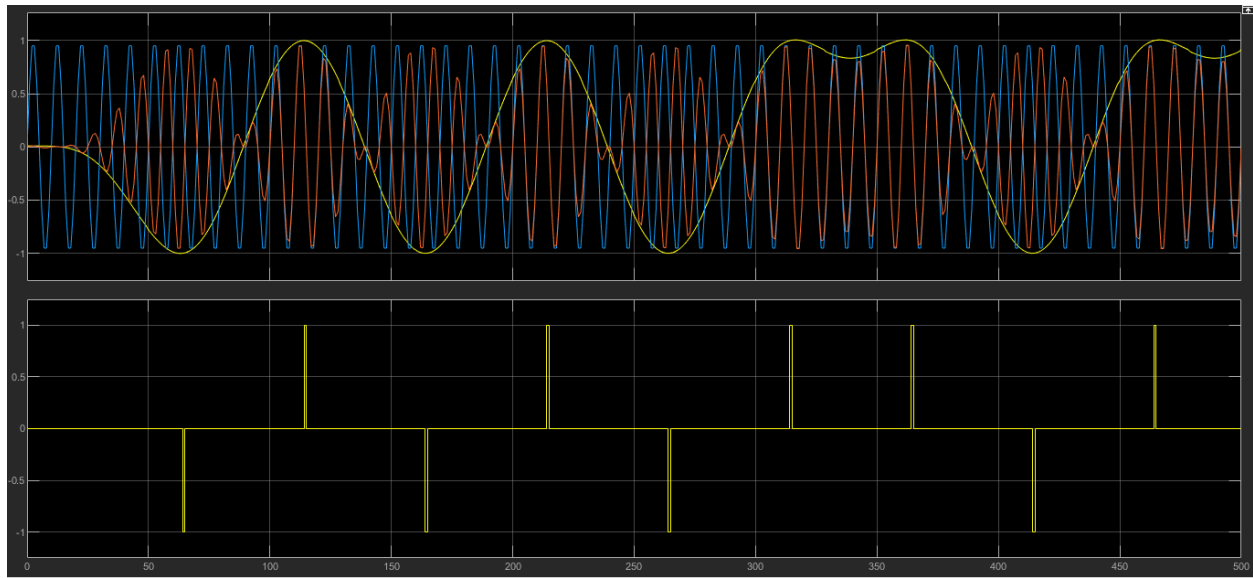


Figure 16: Up-conversion simulation results

The baseband signal is in yellow, the upconverted signal is in red and the unmodulated sine wave is in blue. With BPSK, the signal envelope follows the baseband signal and the carrier is either I phase or out of phase by 180 degrees.

7. QPSK Transmitter:

In contrast to BPSK, QPSK is modelled on 4 symbols. To develop a QPSK, we must first replace the Real Lookup table with a Complex Lookup table. The parameter array in the complex lookup table is the following: $[1 \ j \ -1 \ -j]$. The width of the input is 2. And most importantly, we need to remember that the real and imaginary parts are separated using two carriers: cosine wave for the real part and sine wave for the imaginary part – which need to be handled as such. The final setup looks as follows:

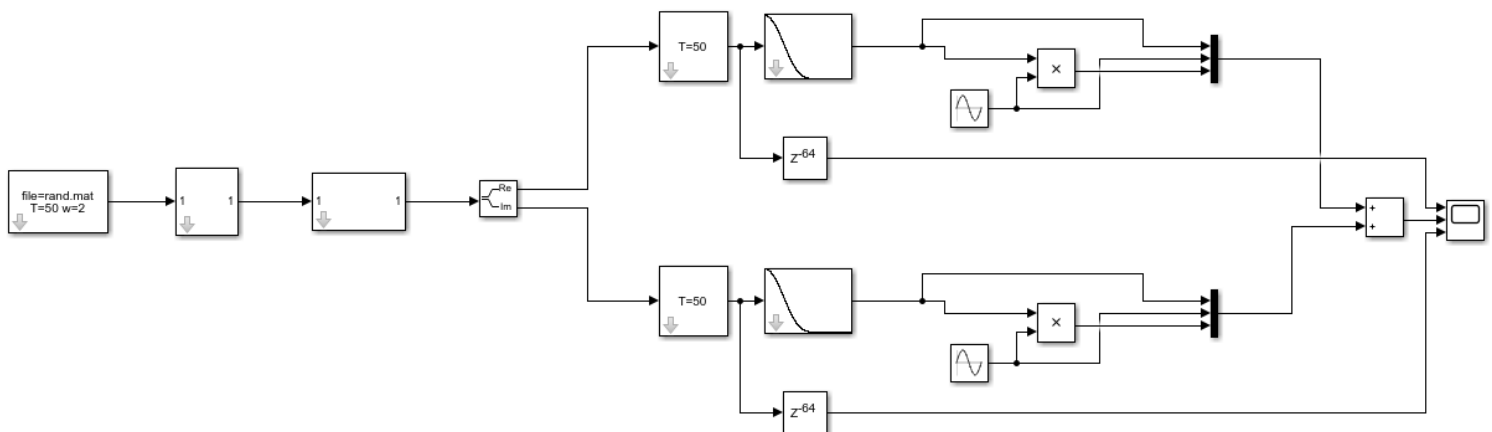


Figure 17: QPSK Transmitter Setup

The results of the simulation have been provided below:

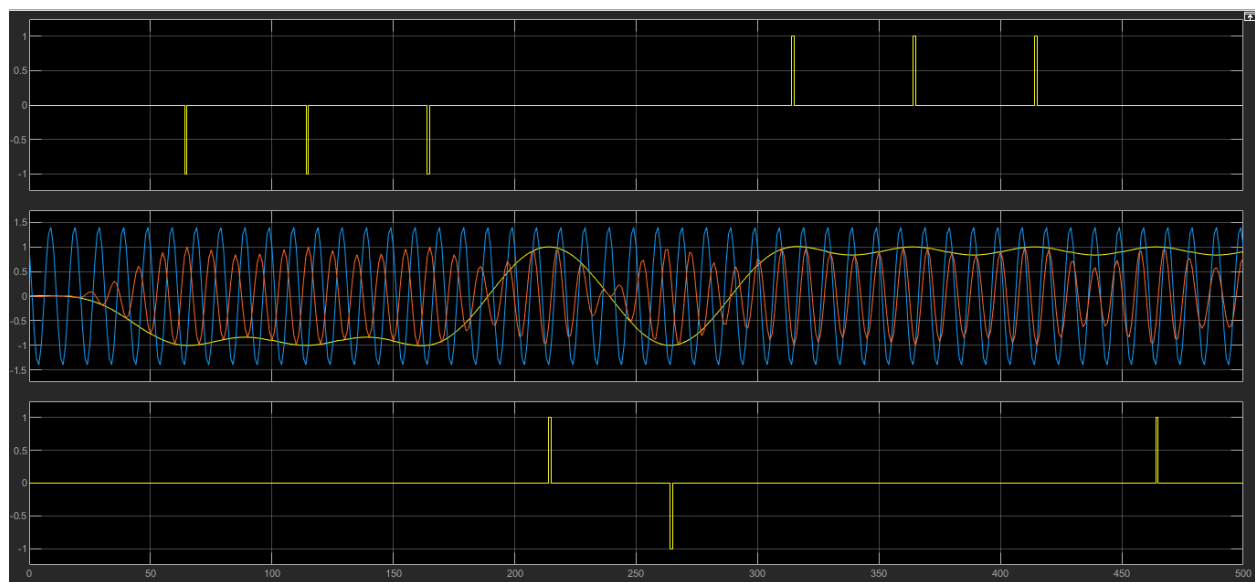


Figure 18: QPSK Transmitter Simulation Results

In the results, we see the incoming waveform, the sampled waveforms and the original stream.

Lab Write-up

1. Explain in one or two paragraphs what you learned about communications systems that you didn't know before.

On this lab, we were introduced to the basic techniques that involve the transmission of a signal. We built a transmitter from scratch, adding a few components along the way until we ended up building two transmitters: BPSK and QPSK. We observed the results of adding each component along the way, which provided us a conceptual understanding of what's happening and how it relates to theory. We learned how single-link communication systems work, and learned about how concepts like bits to symbol mapping, the constellation diagram, pulse shaping, up-conversion etc. come into play. We learned about decoding impulses, RC Filters, Sampling and the importance of delay. We also learned about Inter-Symbol Interference (ISI) and methods to curb it using the methods discussed. Learning about the nuances of Simulink has also added a new arsenal to our skillset, and we were able to learn a new problem solving and simulation technique through the use of block diagrams, that I believe will be absolutely invaluable in the future when dealing with communications systems.

2. Show a picture of your final QPSK transmitter. Explain briefly what the different blocks are for. Explain how you checked that it was working correctly. A plot of the output showing important signals would be very helpful.

The model of the QPSK Transmitter is provided below:

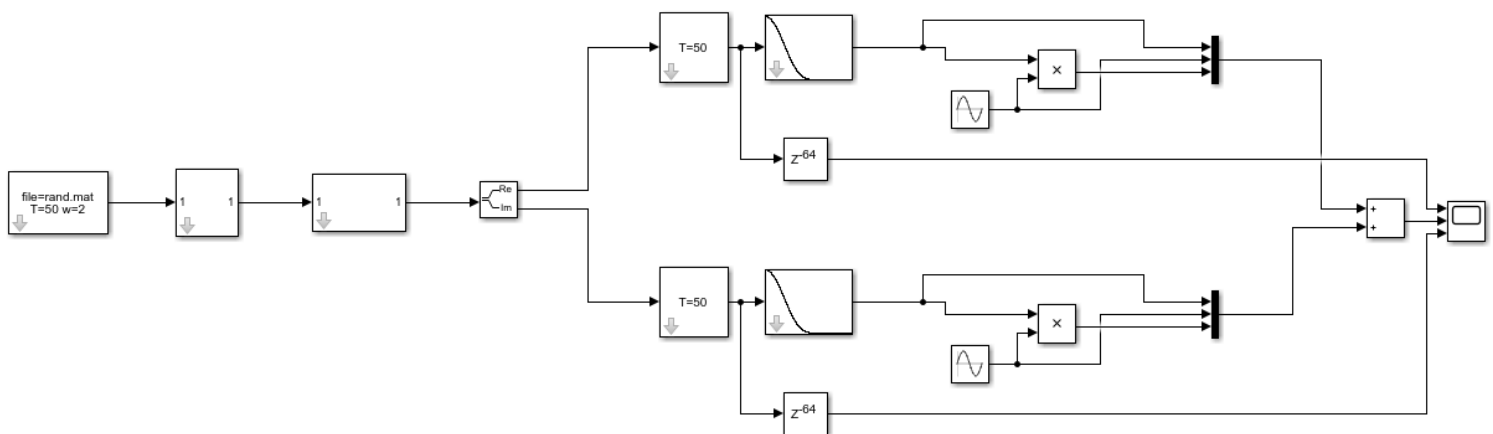


Figure 19: QPSK Transmitter Setup

The results of the simulation have been provided below:

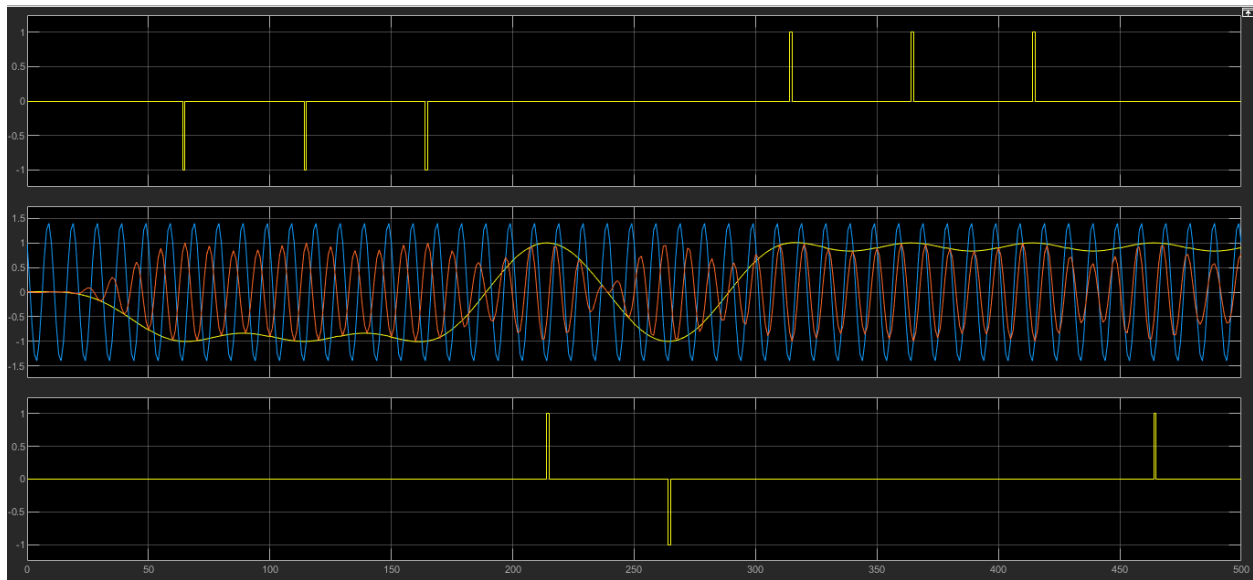


Figure 20: QPSK Transmitter Simulation Results

In the results, we see the incoming waveform, the sampled waveforms and the original stream. The first and the last simulation are the sampled values of the real and imaginary parts. The simulation in the middle firstly contains the combined (real and imaginary) carrier sine wave (shifted by 90 degrees in case of real) in blue, the modulating wave in yellow and the modulated signal in red. The modulating signal is obtained by splitting the result of I/Q mapping (Digital Signal) into real and imaginary parts, sampling them (results shown in first and third graph), low pass filtering the sampled output, and then adding the real and imaginary counterparts.

The modulated signal is obtained by simply multiplying the modulating signal with a sine function (carrier) in time domain (phase shifted by 90 degree in case of real part), and then adding the real and imaginary counterparts.

Consequently, the modulating signal shows peaks at both real and imaginary sample points.

We used different blocks to develop this model. They are as follows:

Input Stream: The input stream block reads binary data from the MATLAB data file 'rand.at', that has a single vector called 'rand' containing 0s and 1s.

Bits to Symbol: The symbol mapper block maps bit 0 to symbol index 0, and bit 1 to symbol index 1.

Complex Lookup 1D: The Complex Lookup table maps a value to each of the parameter values in the array: $[1 \ j \ -1 \ -j]$. It maps 0 to 1, 1 to j , 2 to -1 , and 3 to $-j$.

Complex to Real-Imaginary: This block separates the real and imaginary parts of a function.

Pulse Train: The pulse train samples the input once per period and generates a single unit impulse.

Integer Delay: This block creates a delay.

RC FIR Filter: This block creates a series of pulses from unit impulses. The output's amplitude in this case has a real part and an imaginary part of delayed impulse train.

To validate our results, I firstly made the following modifications:

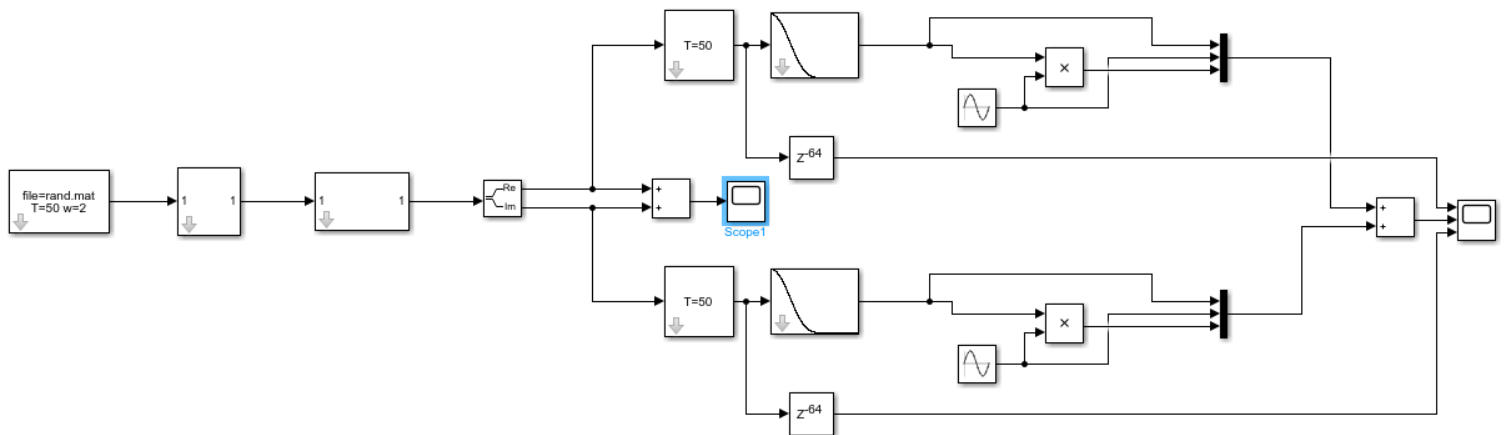


Figure 21: Validation approach

In order to prepare for transmission, we process the real and imaginary parts separately, which are separated at the junction specified above with the “Complex to Real-Imag” block. So the sum the digital waveforms at that juncture should look comparable to the sum at the output, since the output is the sum of the processed waveforms. The sum at the splitting point looks as follows:

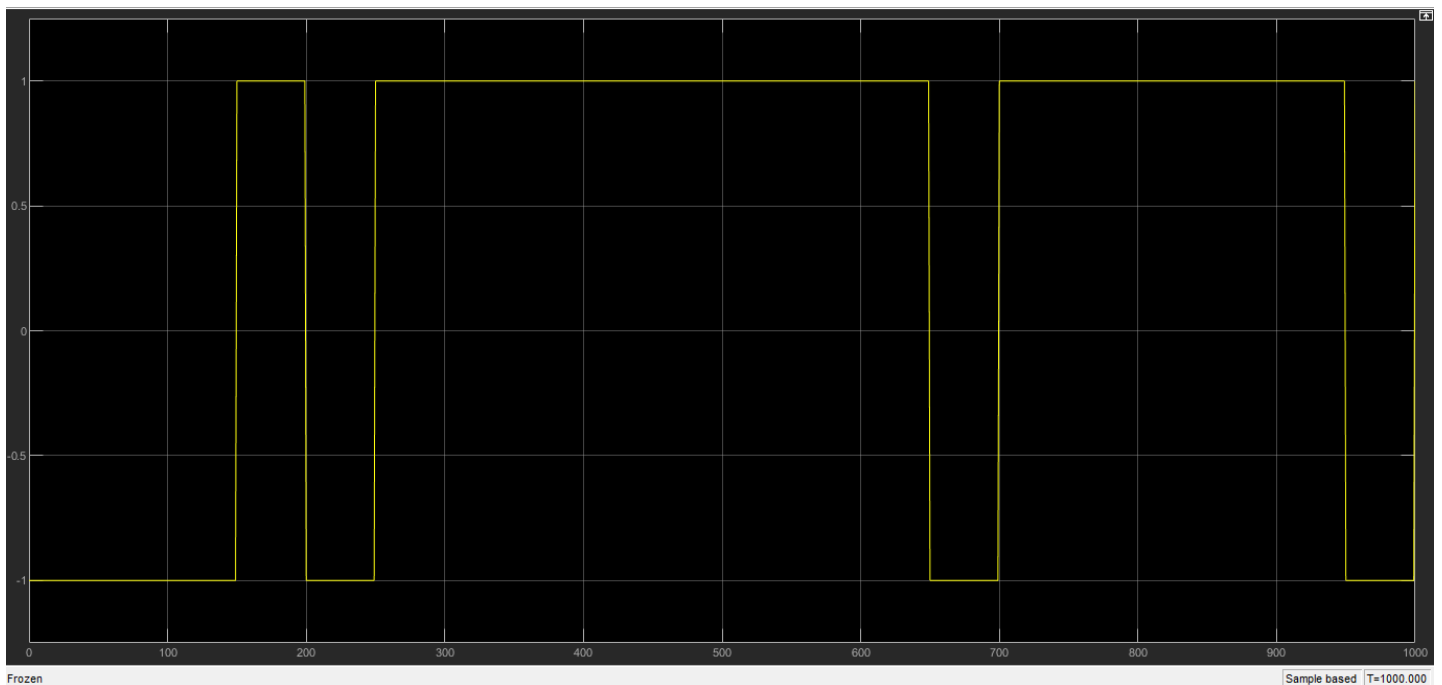


Figure 22: Result of Summation at splitting point, which determines the digital input.

The simulation results at the output looks as follows (Switched to single layout):

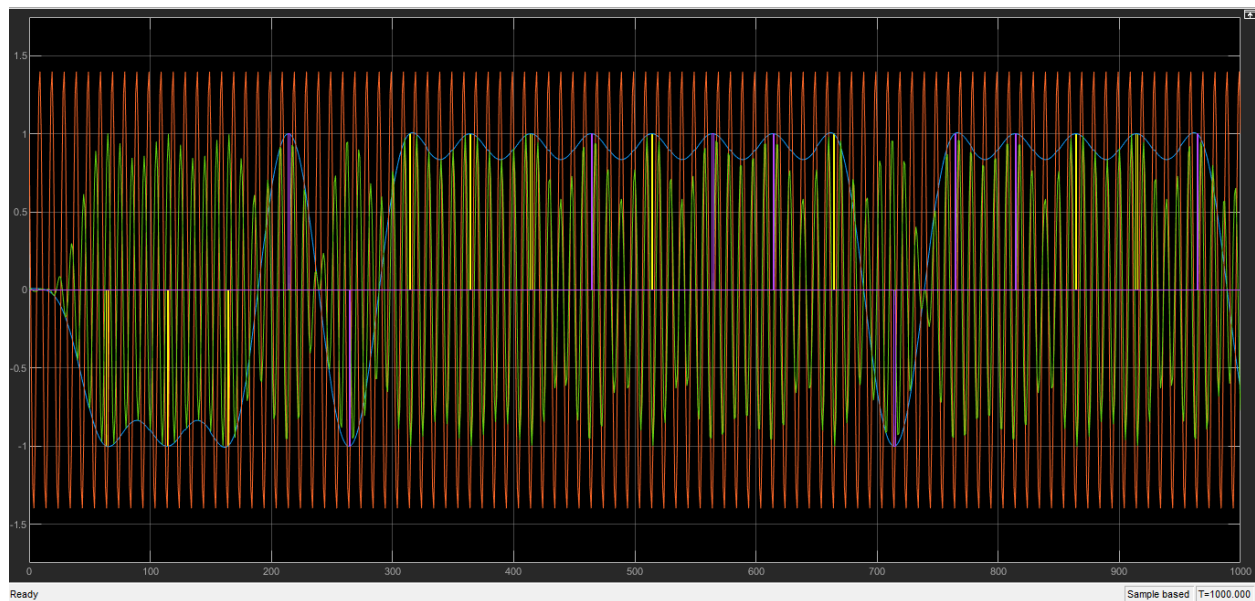


Figure 23: Simulation Results at Output

From the results, we see that the shapes are essentially the same. The final signal has peaks at the sample points, and the sample points are obtained from the digital signal. On a closer look, its like the output is a sinusoid superimposed on the digital input. We could improve the definition of the results by using a higher sampling rate on the input.

3. Explain why do we pulse-shaping in communications systems, rather than just sending rectangular pulses.

By using pulse shaping, we limit the effective bandwidth of transmission. This makes the transmitted signal better suited for communication. By carrying out filtering of transmitted pulses in this way, we effectively manage inter-symbol interference (ISI). As the modulation rate increases, the signal's bandwidth increases: a rectangular wave has larger bandwidth than that of rectangular pulses. Also, when the signal's bandwidth becomes larger than the channel bandwidth, the channel introduces some form of noise or distortion, which appears as ISI. Consequently, through pulse shaping, we ensure that the transmitted signal contains the original, undistorted message by limiting the effective bandwidth.

4. Tell why synchronization (e.g. sampling the received signal at the right place) is critical in a communications system that uses pulse shaping.

During signal transmission, the signal might be subjected to noise or distortion. A possible mismatch can also occur due to a delay element in the transmission system. But, if the signal is sampled at the right position and time, it preserves the original information of the signal and retains the properties of the initial input signal. Otherwise, the result is an output signal that is different from the input signal. This makes synchronization very important. Moreover, synchronization also increases the likelihood of retrieving the original signal.

5. Describe any difficulties you experienced in getting your design to work and how you fixed these problems. I will be really surprised if everything worked perfectly!

A considerable amount of time was lost because I made mistakes while extracting the lab files. It took me some time to figure out which blocks could be found where. Understanding the functions of the blocks also takes some time. It was hard to find the settings to change from time to time. As a result of that, sometimes I would forget to change a setting that would derail all of the simulation results. From that point, backtracking to find what I had missed was an ordeal. It was somewhat difficult to set up the QPSK model also. I don't believe we had a proper understanding of the theory and the functions of all the components by the time we reached that task, so digesting how to suddenly deal with a complex input took some effort. It was also a surprise to figure out the real and imaginary parts would split and be processed separately, even though we knew that from theory. Probably because I wasn't expecting block-based modelling to imitate the theory so closely – as there were many other functional blocks present in the lab work that had no mention in theory.

Conclusion

In this lab, we learned how to make a BPSK and a QPSK transmitter using Simulink. We learned a lot on the nuances involved in developing a transmitter and the different corner cases and sources of error that need to be accounted for. We learned about bit to symbol mapping, Pulse Spectral Density, Spectrally Efficient Pulse Shaping, Decoding Pulses and Up-Conversion, and their implementations. We also learned how to handle complex valued inputs and prepare them for transmission. Best practice for sampling, filtering and modulation was also a part of the lesson. We observed how each of the blocks and slight changes in the parameters manipulated the output simulation results. As a result, we obtained a practical understanding of how signal transmission works, and determined relations between the practical implementation of a transmitter and theoretical foundations of transmission.

References

- DSP and Communication Lab Manual: Transmitter Simulink Study
- <https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/radio-frequency-modulation/digital-phase-modulation-bpsk-qpsk-dqpsk/#:~:text=BPSK%20transfers%20one%20bit%20per,re%20accustomed%20to%20so%20far.&text=QPSK%20is%20a%20modulation%20scheme,we%20need%20four%20phase%20offsets.>